Name: Sahib Bajwa

ID: 107553096

**CSCI 3104, Algorithms** **Profs. Chen & Grochow**

**Problem Set 7 – Due Thur Mar 12 11:55pm** **Spring 2020, CU-Boulder**

---

*Advice 1*: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

*Advice 2*: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

**Instructions for submitting your solutions**:

- All submissions must be typed.
- You should submit your work through the **class Canvas page** only.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please allot at least as many pages per problem (or subproblem) as are allotted in this template.

Quicklinks: 1 2 3 3a 3b 3c 3d

---

Name: Sahib Bajwa
ID: 107553096

**CSCI 3104, Algorithms** **Profs. Chen & Grochow**
**Problem Set 7 – Due Thur Mar 12 11:55pm** **Spring 2020, CU-Boulder**

1. *Let $G(V, E, w)$ be a weighted graph. The* **Cycle Property** *provides conditions for when an edge is* ***not*** *included in any minimum spanning tree (MST) of $G$. The* ***Cycle Property*** *is stated as follows. Let $C$ be a cycle in $G$, and let edge $e = (u, v)$ be a maximum-cost edge on $C$. Then the edge $e$ does not belong to any MST of $G$.*

   *Use an exchange argument to show why this property holds. You may assume that all edge costs are distinct. [**Note:** You may freely use any of the tree properties covered in Michael's Tree Notes on Canvas.]*

   If we remove edge e, which is a maximum-cost edge on cycle C in graph G, we exchange e with another section of cycle C. Since all other edges in cycle C cost less than e, the new tree produced by exchanging e with another section of cycle C will either be a MST of G or closer to becoming a MST of G. Thus, if there is a cycle with an edge of maximum weight e, it cannot be in a MST of G.

Name: Sahib Bajwa
ID: 107553096



2. *Let $G(V, E, w)$ be a connected, weighted graph. A cut of $G$ is a set $C \subset E$ of edges such that removing the edges of $C$ from $G$ disconnects the graph. The* **Cut Property** *provides conditions for when an edge* ***must*** *be included in a minimum spanning tree (MST) of $G$. The* ***Cut Property*** *is stated as follows. Let $C \subset E$ be a cut. Suppose $e = (u, v)$ is the minimum-weight edge in $C$. Then every MST contains the edge $e$.*

   *Use an exchange argument to show why this property holds. You may assume that all edge costs are distinct. [**Note:** You may freely use any of the tree properties covered in Michael's Tree Notes on Canvas.]*

   If we are to remove e from the graph and it is the minimum weight edge in cycle C, then we will have to exchange e with another edge of greater weight in order to complete the tree. Since we are replacing e with an edge of greater weight, the new tree cannot be a MST since it will have a greater weight than the previous tree that included e.

Name: Sahib Bajwa

ID: 107553096

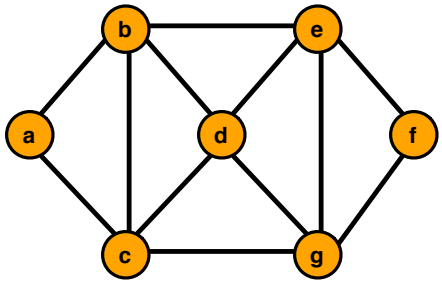
**CSCI 3104, Algorithms** **Profs. Chen & Grochow**

**Problem Set 7 – Due Thur Mar 12 11:55pm** **Spring 2020, CU-Boulder**

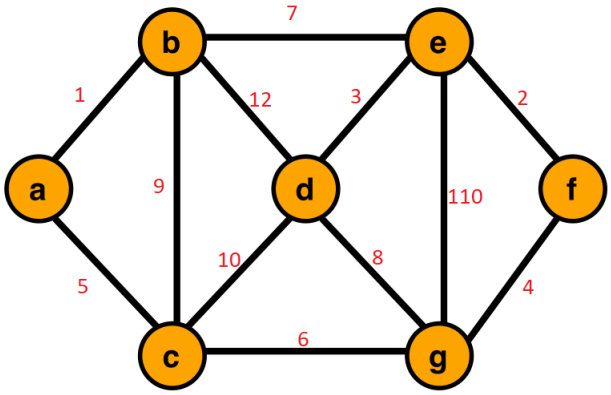
---

3. *Given the following unweighted graph, you're asked to assign a positive integer weight to each edge, such that the following properties are true regarding minimum spanning trees (MST) and single-source shortest path (SSSP) trees:*

   - *All edge weights are distinct*
   - *The MST is distinct from any of the seven SSSP trees. Note that there is a SSSP tree for each node as the source.*
   - *The order in which Prim's algorithm adds the safe edges is different from the order in which Kruskal's algorithm adds them.*

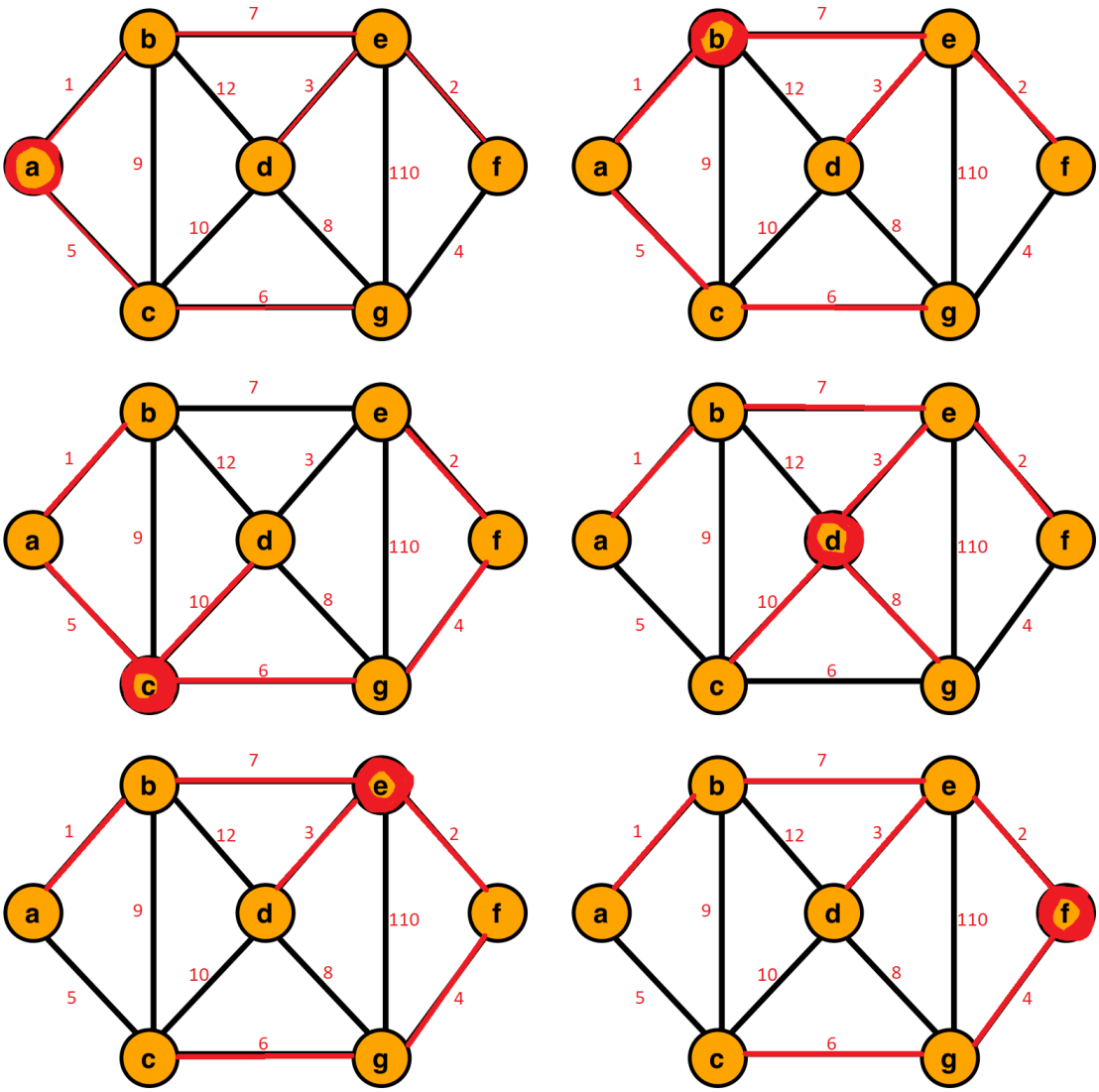


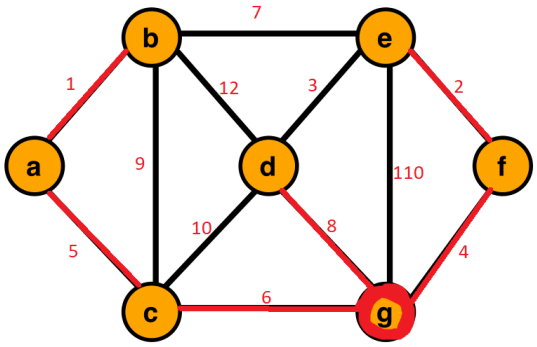

   (a) *Put your picture of the graph (the same one as above)* ***with your edge weights*** *here. Remember that all the edge weights should be distinct:*



(b) *Draw the seven SSSP trees (one for each starting vertex). In each tree, indicate which vertex you used as the starting vertex. For one such tree, clearly articulate the steps taken to select the first three edges using* ***Dijkstra's algorithm****.*

I indicated the starting vertexes by circling the vertex in red. They also go in alphabetical order (left to right, top to bottom).

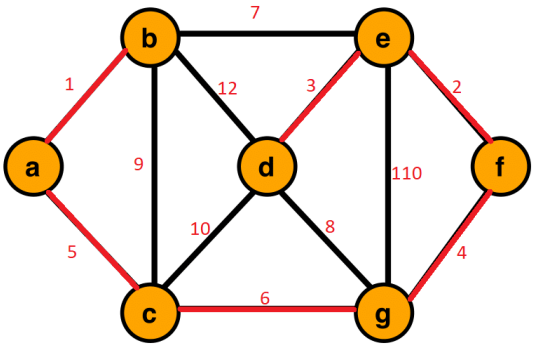





Lets take the last graph, the graph that starts at vertex g, and use it as the example for how Dijkstra's algorithm selects the first 3 edges.

So starting at g, we add edges (g, c), (g, d), (g, e), and (g, f) to our heap. Then we go down the (g, f) endge as it has the lowest weight of the 3 edges in the heap. Adding this edge gets us to add vertex f but does not get us to cut off any edges yet. We then add edge (f, e) to the to the heap, and since the total weight of (g, f) + (f, e) is less than (g, e), we can cut off the edge (g, e). So now we have a total weight of 6 from g to e. The next step we could take would be adding (e,d) to the heap, but since that would make our total current weight higher than either (g, c) or (g, d), we stop that branch at vertex e. So at this point, we take edge (g, c) as or next edge in the algorithm as it has less weight than (g, d). Stopping here, our first three edges that Dijkstra's algorithm has selected are (g, f), (f, e), and (g, c).

(c) *Use Prim's algorithm to compute the MST. List the order in which Prim's algorithm adds the edges, and draw the MST. Furthermore, clearly articulate the steps the algorithm takes as it selects the first three edges.*




I started Prim's algorithm at vertex b. From vertex b, Prim's algorithm will add the edge (b, a) to the heap as it is the smallest weight but also safe. Prim's algorithm will then add edge (a, c) as it is also save and the next edge. We now cut off edge (b, c) from being in the heap at all. Next Prim's algorithm will add edge (c, g) to the heap as it is safe and lower weight than (c, d). So Prim's algorithm will select (b, a), (a, c), and (c, g) as its first three edges.
Overall, Prim's algorithm will add edges in this order:
(b, a), (a, c), (c, g), (g, f), (f, e), (e, g)

(d) *List the order in which Kruskal's algorithm adds the edges. Furthermore, clearly articulate the steps the algorithm takes as it selects the first three edges.*

Kruskal's algorithm adds edges by the order of their weights. Understanding this, the first edge that gets added is (a, b) as it is the smallest weight at 1. Kruskal's algorithm then adds edge (e, f) as it has the next smallest weight at 2. Kruskal's algorithm then adds edge (d, e) as it has the next smallest weight at 3. Thus, the first three edges Kruskal's algorithm adds are (a, b), (e, f), and (d, e).
Overall, Kruskal's algorithm will add edges in this order:
(a, b), (e, f), (d, e), (f, g), (a, c), (c, g)